

2

STR 1.1.2 CDR

AD-A212 978

# NAVAL POSTGRADUATE SCHOOL

## Monterey, California



# THESIS

DTIC  
ELECTE  
OCT 02 1989

S

E

D

A SPECIFICATION OF A CSMA/CD PROTOCOL  
USING SYSTEMS OF COMMUNICATING MACHINES.

by

Mehmet N. LOFCALI

June 1989

Thesis Advisor

G.M. Lundy

Approved for public release; distribution is unlimited.

Unclassified

security classification of this page

# REPORT DOCUMENTATION PAGE

1a Report Security Classification <b>Unclassified</b>			1b Restrictive Markings		
2a Security Classification Authority			3 Distribution/Availability of Report		
2b Declassification/Downgrading Schedule			Approved for public release; distribution is unlimited.		
4 Performing Organization Report Number(s)			5 Monitoring Organization Report Number(s)		
6a Name of Performing Organization Naval Postgraduate School		6b Office Symbol (if applicable) 37	7a Name of Monitoring Organization Naval Postgraduate School		
6c Address (city, state, and ZIP code) Monterey, CA 93943-5000			7b Address (city, state, and ZIP code) Monterey, CA 93943-5000		
8a Name of Funding/Sponsoring Organization		8b Office Symbol (if applicable)	9 Procurement Instrument Identification Number		
8c Address (city, state, and ZIP code)			10 Source of Funding Numbers		
			Program Element No	Project No	Task No
			Work Unit Accession No		
11 Title (include security classification) <b>A SPECIFICATION OF A CSMA/CD PROTOCOL USING SYSTEMS OF COMMUNICATING MACHINES.</b>					
12 Personal Author(s) <b>Mehmet N. LOFCALI</b>					
13a Type of Report Master's Thesis		13b Time Covered From To		14 Date of Report (year, month, day) June 1989	15 Page Count 55
16 Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.					
17 Cosati Codes			18 Subject Terms (continue on reverse if necessary and identify by block number)		
Field	Group	Subgroup	CSMA/CD, communication protocols, formal modeling techniques, systems of communicating machines, local area networks.		
19 Abstract (Continue on reverse if necessary and identify by block number)					
<p>This thesis gives a specification of a communication protocol known as "Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Medium Access Control and Physical Layer Specifications" using <i>Systems of Communicating Machines</i> and shared variables. This protocol is defined in the ANSI/IEEE Standard 802.3 (using the same name). Specification has been analysed using a method called <i>system state analysis</i>. The analysis showed the protocol to be free from deadlocks. The study concludes that CSMA/CD protocol needs a better specification method.</p>					
20 Distribution/Availability of Abstract			21 Abstract Security Classification		
<input checked="" type="checkbox"/> unclassified/unlimited <input type="checkbox"/> same as report <input type="checkbox"/> DTIC users			<b>Unclassified</b>		
22a Name of Responsible Individual G.M. Lundy			22b Telephone (include Area code) (408) 646-2094	22c Office Symbol 52Ln	

Approved for public release; distribution is unlimited.

A Specification of a CSMA/CD Protocol  
Using Systems of Communicating Machines.

by

Mehmet N. LOFCALI  
Lieutenant Junior Grade, Turkish Navy  
BS, Naval Academy/Turkey, 1983

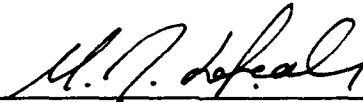
Submitted in partial fulfillment of the  
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL  
June 1989

Author:

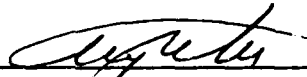


Mehmet N. LOFCALI

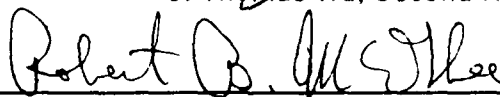
Approved by:



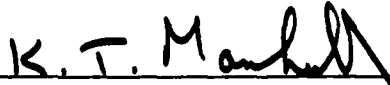
G.M. Lundy, Thesis Advisor



C. Thomas Wu, Second Reader



Robert B. McGhee, Chairman,  
Department of Computer Science



Kneale T. Marshall,  
Dean of Information and Policy Sciences

## ABSTRACT

This thesis gives a specification of a communication protocol known as "Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Medium Access Control and Physical Layer Specifications" using *Systems of Communicating Machines* and shared variables. This protocol is defined in the ANSI/IEEE Standard 802.3 (using the same name). Specification has been analysed using a method called *system state analysis*. The analysis showed the protocol to be free from deadlocks. The study concludes that CSMA/CD protocol needs a better specification method.

Accession For	
NTIS CSMAI	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Avail and/or	
Dist	Special
A-1	



## TABLE OF CONTENTS

I. INTRODUCTION .....	1
A. PURPOSE OF THE STUDY .....	2
B. ORGANIZATION OF THE THESIS .....	4
II. AN OVERVIEW OF PROTOCOL MODELING AND CSMA/CD PROTOCOL	5
A. METHODS CURRENTLY USED IN PROTOCOL MODELING .....	5
1. Communicating Finite State Machines .....	5
a. An Example Specification .....	6
b. Reachability Analysis of the Alternating Bit Protocol .....	8
2. Programming Languages .....	12
3. Extended Finite State Machines .....	13
B. SYSTEMS OF COMMUNICATING MACHINES .....	14
1. The General Model .....	14
2. Specification and Analysis .....	16
C. A REVIEW OF THE IEEE STANDARD 802.3 .....	17
1. Overview .....	17
a. CSMA/CD Operation .....	17
b. Design Issues .....	18
2. Medium Access Control (MAC) Service Specification .....	18
a. Overview of the Service .....	18
b. Service Specification .....	18
3. Media Access Control Frame Structure .....	19
a. General Overview .....	19
b. Frame Format .....	19
c. Invalid MAC Frame .....	20
4. Media Access Control .....	21
a. Overview of Functional model .....	21
b. The Operation of CSMA/CD .....	21

c. Access Interference and Recovery .....	23
d. Relationships to LLC sublayer and Physical Layer .....	23
e. CSMA/CD Access Method Functional Capabilities .....	23
5. CSMA/CD Media Access Control Method .....	24
III. SPECIFICATION OF CSMA/CD PROTOCOL WITH SCM .....	26
IV. SYSTEM STATE ANALYSIS OF THE SYSTEM .....	34
A. ANALYSIS .....	34
B. CALCULATION OF THE NUMBER OF ANALYSIS STATES .....	37
V. CONCLUSIONS AND SUGGESTIONS FOR FURTHER WORK .....	44
LIST OF REFERENCES .....	46
INITIAL DISTRIBUTION LIST .....	48

## LIST OF FIGURES

Figure 1. State Diagram for "Alternating Bit Protocol" . . . . .	9
Figure 2. Reachability Graph for "Alternating Bit Protocol" . . . . .	10
Figure 3. LAN Standart Relationship to the OSI Reference Model . . . . .	22
Figure 4. Specification of the Transmitter . . . . .	29
Figure 5. Specification of the Receiver . . . . .	30
Figure 6. Specification of the Controller . . . . .	31
Figure 7. System State Analysis (continues) . . . . .	38
Figure 8. System State Analysis (continues) . . . . .	39
Figure 9. System State Analysis . . . . .	40
Figure 10. Abstract Modeling of System State Analysis . . . . .	41

## I. INTRODUCTION

When the personal computers became common, the users of these computers needed some machines to support their computing activities such as central databases, departmental databases, printers, plotters and slide makers, electronic mail facilities, access to large machines for numeric\_intensive computing, and access to a supercomputer. The local area network provides a very rich environment to the desktop computer which provides a lot of information and resources to the user of this computer. [Ref. 1]

Computer networks can be classified by[Ref. 2]:

1. Topology:

The way the communicating machines are connected together.

2. Data Transfer Technique:

The way used to process and transmit data.

3. Geographical Coverage:

Classification due to the physical separation of the devices.

Physical separation of the devices defines three types of networks:

1. Wide Area Network (WAN):

Use of todays public telecommunication facilities has been provided to the users with access to the processing capabilities and data storage capabilities of large mainframes and also allowed fast data interchange between the members of the network [Ref. 1: p.3]. such kind of networks which are physicaly distributed more than 10 Km are called wide area network [Ref. 3: p.6].

2. Metropolitan Area Network (MAN):

These kind of networks take place between wide area networks and local area networks. As it is implied by it's name these are city wide networks where physical seperation is between 1 Km to 50 Km. [Ref. 1: p.4]

3. Local Area Network (LAN):

The IEEE defines a LAN as follows: *A datacomm system allowing a number of independent devices to communicate directly with each other,*



*within a moderately sized geographic area over a physical communications channel of moderate data rates. The physical separation range is between a few feet and 5 Km [Ref. 1: p.4].*

LANs can also be divided into subgroups depending the topology used : *Bus/tree, star, and ring*. The bus or tree topology is characterized by the use of a multipoint medium. The bus is simply a special case of a tree, with only one trunk and no branches. Since all the devices share a common transmission medium, only one device can use this medium at a time. The distributed medium access protocol is used to determine the next transmitting station. Each station in the system monitors the medium and copies the message if it is addressed to itself. [Ref. 4: p.333]

Although there have been a number of *medium access control* (MAC) techniques proposed for bus topology, the Ethernet's CSMA/CD ( *Carrier Sense Multiple Access With Collision Detection* ) has become the most popular one and it is the one which has been selected for standardization by IEEE 802 Local Network Standards Committee. [Ref. 1]

#### **A. PURPOSE OF THE STUDY**

In order for data communications to take place, a set of requirements has to be fulfilled. There should be some kind of media, through which the signals can pass, and the messages that propagate through this medium should be encapsulated. Thus the message should be encoded to travel in this medium as a signal, and when this signal received it should be decoded and reformed as a message. However this process is error prone and it should be prepared carefully. To protect the messages from errors requires detailed rules and algorithms or some kind of special language between the sender and receiver. These algorithms are called *communication protocols* [Ref. 5: p.2]. The essence of protocols is to ensure that pieces of the system work as a harmonious whole [Ref. 6: p.46].

These protocols should also support the LAN to handle the following requirements which are stated by Ethernet [Ref. 1]:

1. Data rate between 1 and 10 megabits per second (Today this upper limit is about 100 megabits per second.)
2. Geographic range should be about 1 Km
3. Support for several independent devices
4. Use of simplest possible mechanisms to function properly
5. Reliability
6. Minimal dependency to any centralized control
7. Efficient use of resources and network
8. Stability under high loads
9. Easy installation and expandability
10. Ease of maintenance
11. Low cost

Considering the increasing size of today's networks, complexity of the network elements, and the variety of existing and expected products it is clear that network systems need correct, clear, unambiguous and expandable protocols [Ref. 5: p.2] in order to reach the goals stated above. Protocol specification and analysis techniques have been a major research subject [Ref. 6: p.51].

While CSMA/CD protocol has handled most of the LAN requirements, the research has shown that CSMA/CD is the most load sensitive LAN protocol. Thus longer delays may be expected under heavy work loads because of the increasing number of collisions. [Ref. 4: p.367]

Increasing importance of expandability in computer science from hardware to software should be expected from LANs without decreasing efficiency with the added stations.

An understanding of the importance of CSMA/CD protocol in computer networks has encouraged the author to study formal protocol specification techniques. The purpose of this thesis is to make a formal specification of the CSMA/CD protocol as written in the "ANSI/IEEE Standard 802.3, Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and

Physical Layer Specifications," using systems of communicating machines and to analyze it by using system state analysis.

## **B. ORGANIZATION OF THE THESIS**

This chapter serves as an introduction.

**Chapter II, Background:** provides an overview of protocol specification and analysis techniques including an emphasis on Communicating Finite State Machines and Extended Finite State Machines. A general review of the IEEE Standard 802.3-1985 Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications is also given.

**Chapter III, Specification of IEEE Standard 802.3:** discusses the specification of the CSMA/CD protocol with systems of communicating machines and shared variables.

**Chapter IV, Analysis of the protocol:** provides the analysis of the CSMA/CD protocol modeled with systems of communicating machines.

**Chapter V, Conclusions and Recommendations:** summarizes the conclusions reached by this study along with suggestions for further work.

## II. AN OVERVIEW OF PROTOCOL MODELING AND CSMA/CD PROTOCOL

This chapter discusses the main protocol modeling methods and provides an overview of the CSMA/CD, emphasizing the MAC (Medium Access Control) layer of the protocol.

### A. METHODS CURRENTLY USED IN PROTOCOL MODELING

Although there have been several formal protocol modeling methods [Ref. 6] they can be categorized into one of the following:

1. Communicating Finite State Machines (CFSM),
2. Programming Languages,
3. Extended Finite State Machines (EFSM),

#### 1. Communicating Finite State Machines

The CFSM models are easy to analyze, and the protocol can be analyzed by using *reachability analysis* and checked for correctness. Protocols specified by CFSM models are also simple and easy to understand.

In a CFMS model, each process is specified as a finite state machine. The protocol system is a set of machines:

$$M = [m_1, m_2, \dots, m_n]$$

where there is a first-in-first-out (FIFO) queue in both directions, which represents the communication channel, between each pair of machines. A machine is specified as a set of states, a set of transitions, and a mapping between the states and transitions. The transition types include a *send-transition*, a *receive-transition*, and an *internal-transition*. While an internal transition does not change the contents of any queue, a receive-transition takes the message from the in front of an incoming queue, and the send-transition places the message at the end of an outgoing queue. In order for a transition to take place, certain conditions must hold. For

example, for a receive-transition the message to be received should be present at the head of the incoming queue.

The protocol is defined with a diagram, which is generally called a *state-transition diagram* (or simply, *state diagram*). The states are given names or numbers, and are usually shown as circles. The possible transitions between states are indicated by pointed arcs with the transition stated alongside the arc [ Ref. 3: p.118 ]. In the simplest form, the transition with a "-" (minus) sign points an outgoing transition and a transition with a "+" ( plus) sign points an incoming transition.

**a. An Example Specification**

As an example, the simple flow control method known as "alternating bit protocol" can be defined using a CFSM model.

The alternating bit protocol works as follows. There are two "machines;" the sender, and the receiver and two communication channels between them in both direction. Initially they are in a "ready" state. When the sender has a message to send, it inserts the frame to the outgoing queue and moves to a second state where it awaits an *acknowledgment from the receiver*. The receiver next receives the message from the queue and moves to a second state, from there it sends an acknowledgement and changes its state. The sender in turn receives the acknowledgment from the incoming queue and changes its state. And this scheme goes on until they both reach their initial states.[Ref. 7]

The CFSM model of this scheme would be defined as a protocol machine  $PM = (S, M, I, T, C)$ , where,

$S = \text{sets of states of machines} = \{S_1, S_2\}$

$S_1 = \text{states of } m_1 = \{0, 1, 2, 3\}$

$S_2 = \text{states of } m_2 = \{0, 1, 2, 3\}$

$M = \text{sets of messages} = \{M_1, M_{21}\}$

$M_{12}$  = messages that can be sent from  $m_1$  to  $m_2 = \{D0, D1\}$

$M_{21}$  = messages that can be sent from  $m_2$  to  $m_1 = \{A0, A1\}$

$I$  = set of initial states of machines =  $\{I_1, I_2\}$

$I_1$  = initial state of  $m_1 = 0$

$I_2$  = initial state of  $m_2 = 0$

$T$  = partial transition function

$S_i \times \Sigma_i \mapsto S_j$ , where,

$$\Sigma_i = \{-x \mid x \in M_{ij}\} \cup \{+y \mid y \in M_{ji}\} \quad i, j = 1, 2$$

and,

$-x$  = sending of message  $x$

$+y$  = receiving of message  $y$

for  $m_1$

for  $m_2$

$$0 \times -D0 \mapsto 1$$

$$0 \times +D0 \mapsto 1$$

$$1 \times +A1 \mapsto 2$$

$$1 \times -A1 \mapsto 2$$

$$2 \times -D1 \mapsto 3$$

$$2 \times +D1 \mapsto 3$$

$$3 \times +A0 \mapsto 0$$

$$3 \times -A0 \mapsto 0$$

$C$  = communication channels =  $\{C_{12}, C_{21}\}$

$C_{12}$  = The FIFO queue between  $m_1$  and  $m_2$

$C_{21}$  = The FIFO queue between  $m_2$  and  $m_1$

where, the contents of the queues are  $c_{ij}$  ( $c_{12} \in M_{12}$  ,  $c_{21} \in M_{21}$ ) [Ref. 8].

This definition is then illustrated as seen in Figure 1 on page 9, where  $m_1$  and  $m_2$  are shown as finite state machines with the communication channels between them. This is a simple model which is provided as an example and does not deal with some details such as lost messages.

**b. Reachability Analysis of the Alternating Bit Protocol**

Reachability analysis is a common method used for analyzing the CFSM models. In this method the analysis is done by generating all possible *global states* from the initial global state. A global state is a tuple consisting of the states of each machine and the contents of each queue in the system. For the specification in Figure 1, this would be a 4-tuple,

$$\langle (S_1, S_2), (Q_1, Q_2) \rangle$$

where,

$$S_1 = \text{state of } m_1$$

$$S_2 = \text{state of } m_2$$

$$Q_1 = \text{contents of the queue } C_{12}$$

$$Q_2 = \text{contents of the queue } C_{21}$$

The analysis starts with the initial global state, and the reachability graph is constructed by writing down the next possible global state(s) with an arc labeled with the transition which leads to that state. Figure 2 on page 10 shows the reachability graph for the "Alternating Bit Protocol".

The graph is generated as follows: in this example initially both machines ( $m_1$  ,  $m_2$ ) are in state 0 and the queues ( $C_{12}$  ,  $C_{21}$ ) are empty.

$$\langle (0, 0), (E, E) \rangle$$

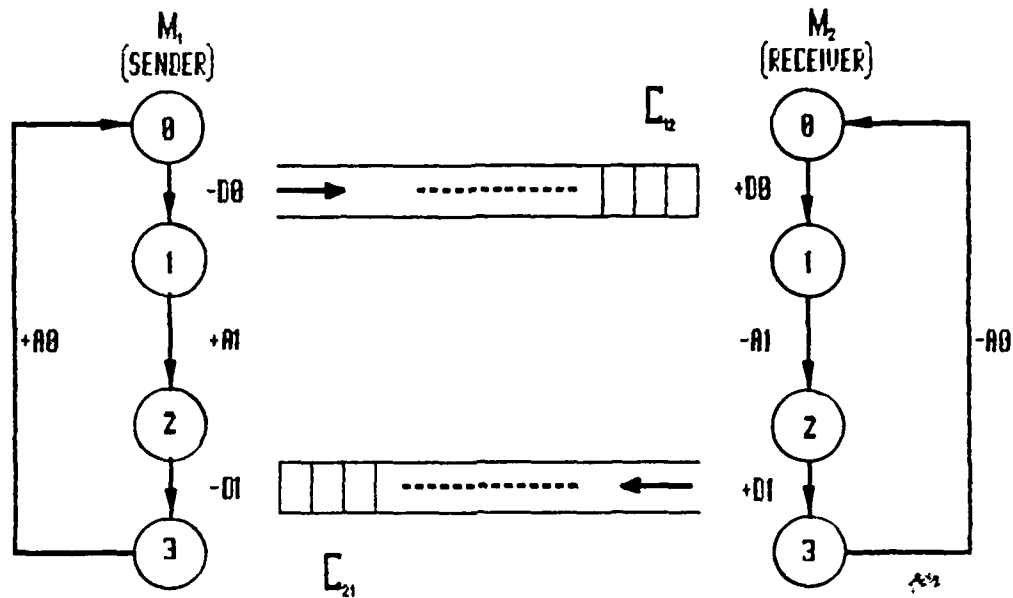


Figure 1. State Diagram for "Alternating Bit Protocol"

Inspecting the FSMs, there are two transitions that may take place.  $m_1$  may send a D0 or  $m_2$  may receive a D0. Since the queues are empty (E), the receive-transition is not possible. The only possible transition is "-D0." Thus  $m_1$  puts D0 in  $C_{12}$  and moves to state 1.

$$<(1, 0), (D0, E)>$$

Again, two transitions are possible from this global state.  $m_1$  may receive an A0 or the  $m_2$  may receive a D0. Since  $C_{21}$  is empty,  $m_1$  can not make the receive-transition. Thus, the  $m_2$  receives D0 from the  $C_{12}$  and changes its state to state 1.

$$<(1, 1), (E, E)>$$



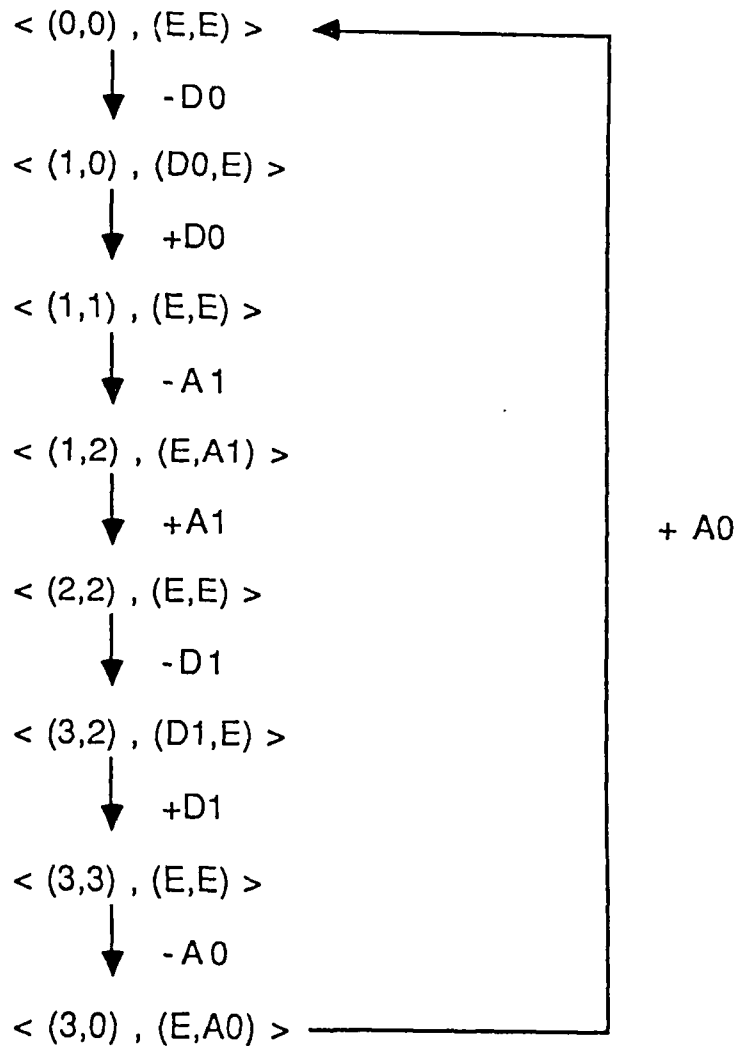


Figure 2. Reachability Graph for "Alternating Bit Protocol"

Now there are two possible transitions here again.  $m_1$  may receive an A1 and move to state 2 or  $m_2$  may send an A1 and change its state to state 2. However, since  $C_{21}$  is empty  $m_1$  can not receive and  $m_2$  moves to state 2 copying A1 in  $C_{21}$ .

$\langle (1,2), (E,A1) \rangle$

The next transition from here will be a "+A1" which moves  $m_1$  from state 1 to state 2.

$$<(2, 2), (E, E)>$$

At this point since the queues are empty the only possible transition is "-D1" which moves  $m_1$  to state 3.

$$<(3, 2), (D1, E)>$$

Similarly, the only possible transition is by  $m_2$  by taking D1 from  $C_{12}$  and  $m_2$  moves to state 3.

$$<(3, 3), (E, E)>$$

Here again, since both queues are empty only  $m_2$  can put a message (A0) in  $C_{21}$  and goes back to the state 0.

$$<(3, 0), (E, A0)>$$

The next transition from here will be the "+A0", which takes  $m_1$  to the state 0.

$$<(0, 0), (E, E)>$$

This completes the analysis since both machines are in their initial global states. When  $m_1$  wants to send another message, the same sequence will happen again following a cyclic behavior.

The purpose of the reachability analysis is to find out if certain types of errors in the system are possible. A successful reachability analysis for a certain protocol shows that the protocol is free from following types of errors:

1. Deadlock:

A global state where both machines are in receiving position and all the queues are empty. Since none of the machines can move to another state, deadlock causes an unexpected stop in the analysis.

2. Unspecified reception:

This is a state where one or more machines are in a receiving state but the message at the head of the incoming queue is not the message to be received.

3. Non-executable transition:

This is a transition specified in the state-diagram, which may never be executed by the protocol. These type of errors are harmless and they could be a design error or they could be placed in the system by the designer for debugging purposes during the design phase. After the analysis is done they can be eliminated or converted to executable transitions.

Although simple and easy to understand, the Communicating Finite State Machines are not without following drawbacks [Ref. 7]:

1. Undecidability problem:

- a) In most general cases, it is undecidable whether the analysis will ever terminate.
- b) Even it is decidable on some cases, it is still uncertain on some errors.

2. State explosion problem:

- a) Even if reachability graph is finite, it may be too large to generate.
  - b) Specification can have too many states.
- It is difficult to deal with timing and errors.

3. The model assumes channels to be of infinite length and that no messages are ever lost; this is unrealistic for modeling of some protocols.

## 2. Programming Languages

Methods using programming languages are more powerful than CFSM models in that they are very close to actual implementation. They also give the ability to model any protocol. However, they are more complex and difficult to understand. As a result, the analysis of the protocol is more difficult. [Ref. 5 : p.10]

Communicating Sequential Processes (CSP), Language of Temporal Ordering Specification (LOTOS), Protocol Description and Implementation Language (PDIL) are some examples of languages used for protocol specification and modeling. ADA has also been suggested for possible use as a protocol specification language since it supports parallel programming [Ref. 9].

### **3. Extended Finite State Machines**

The disadvantages of the CFSM model and languages have encouraged scientists to seek with improved protocol specification techniques. One of these is generally called "Extended Finite State Machines" (EFSM). While CFSM model has no memory, EFSM model utilizes variables which are used to store less important information such as sequence numbers and addresses. [Ref. 6]

One of the EFSM method is *Specification and Description language* (SDL) which has been prepared by *The Consultative Committee for Internal Telephone and Telegraph* (CCITT). This model has been proven as a successful model. However since it is hard to implement, it requires some refinements.

The *Parallel Activity Specification Scheme* (PASS) models the protocol by using PROLOG language and the machines in the network are modeled as extended finite state machines with local variables. [Ref. 10 ]

In *Extended State Transition Language* (ESTL) (also called "Estelle"). the protocol is modeled as a set of modules communicating with each other. The modules are specified as extended finite state machines by means of an extended PASCAL language. [Ref. 11 ]

The technique used in the *Token Ring Protocol* standard [Ref. 12]. is also an FSM method. The protocol is specified by use of extended finite state machines, tables, and descriptive text [Ref. 2].

One other extended finite state machines model is the *Systems of Communicating Machines* (SCM) which this thesis focused on, has been proposed by Lundy and Miller[Ref. 13] as an extended finite state machine,

and this model attempts to "find a balance" between the two extremes, the CFSM and the protocol languages [Ref. 5] removing their disadvantages.

## B. SYSTEMS OF COMMUNICATING MACHINES

The SCM model uses extended finite state machines for both machines and channels. In this system machines can communicate only through shared variables, and local variables of any machine can be accessed only by this machine. Any transition is associated with *enabling predicates* and *actions*. Unlike CFSM this model has no queues to represent channels. This model uses the communication channel as a shared variable or a machine which shares variables with the machines it is connecting. This allows more precise control over the behavior of the channel and allows errors to be modeled more explicitly. [Ref. 5]

Like CFSM, this model is defined with a diagram which is called *state diagram*, and the states are given names and are shown as circles. The transitions between states are labeled but are not signed.

### 1. The General Model

The general model *systems of communicating machines* might be used to model some other types of systems such as operating systems and parallel programs. Then with some placed restrictions, this model can be used to model protocols.[Ref. 14]

A *system of communicating machines* is an ordered pair  $C = (M, V)$ , where  $M = [m_1, m_2, m_3, \dots, m_n]$  is a finite set of *machines*, and  $V = [v_1, v_2, v_3, \dots, v_n]$  is a finite set of shared variables. Each machine  $m_i$  has two designated subsets of shared variables which are called  $R_i$  and  $W_i$ . The subset  $R_i$  of  $V$  is called the set of *read access variables* for machine  $m_i$ , and the subset  $W_i$  the set of *write access variables* for machine  $m_i$ .

The CFSM model defines each machine by a tuple:

$$(S_i, s_i, L_i, N_i, \tau_i)$$

where

$S_j$  = finite set of states;

$s \in S_j$  = initial state of  $m_j$

$L_j$  = finite set of local variables;

$N_j$  = finite set of names, each of which is associated with a unique pair  $(p, a)$ , where  $p$  is a predicate on the variables of  $L_j \cup R_j$ , and  $a$  is an action on the variables of  $L_j \cup R_j \cup W_j$ . Specifically, an action is a partial function  $a: L_j \times R_j \rightarrow L_j \times W_j$  from the local variables and read access variables to the local variables and write access variables.

$\tau_j: S_j \times N_j \rightarrow S_j$  is a transition function, which is a partial function from the states and the names of  $m_j$  to the states of  $m_j$ .

The machines of this model are the processes and channels of a protocol system, and the processes of a protocol system communicates through the shared variables. Predicates determines the transitions to be done, and a transition can be taken only the predicate for that transition is true. That transition executes the action associated with that name, and the action changes the values of local and shared variables allowing other predicates to become true.

Each local variable has a name and range which is determined in the specification. The range of a variable may be between an empty value (denoted by 0) and finite or countably infinite value. It is also allowed to be left "undefined" value for analysis purposes.

The transition  $\tau_j$  of machine  $m_j$  is enabled when the enabling predicate  $p$ , associated with name  $n$ , is true, and it is executed if the predicate  $p$  is true and the machine  $m_j$  is in state  $s_j$ , ( $j, k \in S_j$ ) which is defined in partial transition function  $\tau(s_j, n) = s_k$ . Since the execution of this transition changes the state of  $m_j$  from state  $s_j$  to  $s_k$ , and the action  $a$  associated with  $n$  occur simultaneously, the execution is called as atomic transition.

The definition of transition function is important since it allows simultaneous transitions, thus two different machines ( $m_i$  and  $m_j$ ) may make their transitions at the same time if their enabling predicates are true. This

situation can be included in system state analysis and does not cause any error in the system unless the simultaneous actions activated by  $T_i$  and  $T_j$  are to write into some variable  $V$  which is shared by the two machines. This situation may be resulted in two ways such as:

1. Only one of the machines may succeed to write into this shared variable. This action does not effect the protocol system.
2. Two of the machines may succeed to write into the shared variable. This action leaves shared variable  $V$  with garbled message in it.

The second result considered as an *ill-defined transition* and raises the question of whether or not a given system is ill-defined. This question brings up the following theorem:

*Theorem : It is undecidable whether an arbitrary system of communicating machines is ill-defined (proof can be found in [Ref. 5]).*

Ill-defined transitions may decrease the system efficiency. Use of some special types of variables can prevent ill-defined transitions.

## **2. Specification and Analysis**

The analysis proposed for SCM is similar to reachability analysis, and is referred to as "system state analysis". The system state analysis is used to analyze the protocol specified and assures that the protocol modeling is free from some kind of errors. The errors which can be found with system state analysis are same with the errors found with the reachability analysis of the CFSM except some definition differences [Ref. 5].

A system is said to be *deadlocked* if every machine  $m_i$  is in a state  $X_i$  and none of the transitions out of state  $X_i$  is enabled and the state is not a final state.

This definition also includes some of the unspecified receptions as well since in SCM the channel is defined as an explicit machine. Because of this explicit definition, an empty channel and a channel containing an unexpected message are both different state tuples, and the unspecified receptions may not cause the system to be halted (deadlocked).

The definition of nonexecutable transition is same as in CFSM and covers the transitions which can never be executed from the initial system state.

The states used in state analysis is defined as follows:

1. *system state tuple* is a vector of all machine states.
2. *system state* is a tuple of all machine states together with the enabled outgoing transitions.
3. *global state* of a system consists of the system state tuple, plus the values of all variables, both local and shared.

Two system states are called *equivalent* if every machine is in the same state, and the same outgoing transitions are enabled.

### C. A REVIEW OF THE IEEE STANDARD 802.3

The discussion, figures and tables provided in this part has been taken from *ANSI/IEEE Standard 802.3-1985, Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications* [Ref. 15]. The details which are unnecessary for the purposes of this thesis are omitted when appropriate.

#### 1. Overview

##### a. CSMA/CD Operation

The CSMA/CD media access method is the means by which two or more stations share a common bus transmission medium. A transmitting station first waits for a quiet period on the medium and determines whether or not another station is transmitting a message. If the transmission medium is available, then it sends its message in bit-serial form. When two or more stations all have messages to send occasionally, they may send their messages simultaneously, which results with a *collision*. Since the result of collision leaves a garbled message in the medium, receiving stations ignore the garbled message. The transmitting stations intentionally sends a few additional bytes to ensure propagation of the collision throughout the system



and then stops transmitting. After the collision they wait for a random amount of time and then they attempt to transmit again.

**b. Design Issues**

Local area networks can be designed basically in two ways:

1. *Implementation*: Focuses on the actual components, their packaging and interconnection issues.
2. *Architecture*: Emphasize the logical divisions of the system and how they fit together.

The architectural design is used in this protocol since this design has *clarity* and *flexibility* feasibilities. Under architectural design this system is separated in two main parts as: the Media Access Control (MAC) sublayer of the Data Link Layer, and the Physical Layer. This partition allows the system to be compatible with the various kinds of implementations.

**2. Medium Access Control (MAC) Service Specification**

**a. Overview of the Service**

The services provided by the MAC sublayer allow the local Logical Link Control LLC sublayer entity to exchange LLC data units with peer LLC sublayer entities. The services are described in an abstract way and do not imply any particular implementation. Thus there may not be one-to-one correspondence between the primitives and the implementations.

**b. Service Specification**

The service specification of the interface between the LLC sublayer and the MAC sublayer defines three service primitives:

**1. MA\_DATA.request:**

This primitive defines the transfer of data from a local LLC sublayer entity to a single peer LLC entity or multiple peer LLC entities in case of group addresses. It defines the address of either a single or a group MAC entity address, sufficient information about the length of the data unit which will be transmitted by the MAC sublayer, and the quality of the service requested by the LLC or a higher layer. When the MAC layer receives this primitive it appends all specific MAC fields, including *destination address* (DA), *source address*, (SA) and any fields unique to the particular media access method.

2. MA\_DATA.confirm:

This primitive provides an appropriate response to the LLC sublayer MA\_DATA.request primitive, and indicates the success or failure of the previous associated MA\_DATA.request.

3. MA\_DATA.indication:

This primitive is passed from the MAC sublayer entity to the LLC sublayer entity or entities in the case of group addresses. It indicates the arrival of a data frame. The data frame is reported to the LLC sublayer, only if they are validly formed, received without error, and their destination address is the local MAC entity. It defines the destination address or addresses (in case of group address) of incoming data frame, the individual source address of incoming frame, the length of the data unit, and the status information.

### 3. Media Access Control Frame Structure

#### a. General Overview

The CSMA/CD standard defines a specific frame structure which is used to transmit throughout the network. The frame has eight fields, which are at the fixed sizes except the LLC data and PAD fields. The sizes of LLC data and PAD fields is flexible in the limits of minimum and maximum values of number of bytes, which is determined by the specific implementation of the CSMA/CD MAC sublayer.

#### b. Frame Format

1. *Preamble Field*: The frame begins with preamble. It includes 7 bytes of synchronization format, which synchronizes the receiving station with received frame timing. The preamble pattern is 7 bytes of 10101010.
2. *Start Frame Delimiter Field*: This field is 1 byte of 10101011. It immediately follows the preamble field and indicates the start of a frame.
3. *Address Fields*: The MAC frame contains two address fields: the *Destination Address Field* and the *Source Address Field*, in that order. The Destination Address field shall specify the destination address or addresses for which the frame is intended. The Source Address field shall identify the station from which the frame was initiated. The representation of each address field shall be as follows:
  - a. Each address field contains either 2 bytes (16 bits) or 6 bytes (48 bits). However, at any given time, the Source and Destination Address size should be the same for all stations on a particular local area network. The support of 2 byte or 6 byte address field has been left to the manufacturer.

- b. The first bit of the Destination Address field is used as an address type designation bit to identify the Destination Address as an individual or as a group address. In the Source Address field, the first bit is reserved and set to 0. If this bit is 0, it indicates that the address field contains an individual address. If this bit is 1 it indicates that the address field contains a group address that identifies none, one or more, or all of the stations connected to the local area network.
  - c. For 6 byte addresses, the second bit is used to distinguish between locally or globally administered addresses. For globally administered addresses, the bit is set to 0. If an address is to be assigned locally, this bit is set to 1.
4. *Length Field*: The Length field is used to indicate the number of data bytes in the LLC data field. If the number of data byte is less than the minimum number of data bytes required for proper operation of the protocol, a PAD field (a sequence of bytes) will be added at the end of the data field but prior to the FCS field, specified below. The size of the Length field is 2 bytes.
  5. *Data and PAD Fields*: The data frame contains a sequence of  $n$  bytes. A minimum number of data bytes is required for correct operation of CSMA/CD protocol and this minimum number is specified by the particular implementation of the standard. If necessary the data field is extended by appending extra bits (PAD) in units of bytes after the LLC data field but prior to calculating and appending the FCS. The size of a PAD is determined by the size of the data field supplied by LLC and the minimum frame size and address size of the particular implementation. The maximum size of the data frame is determined by the maximum frame size and address size parameters of a particular implementation.
  6. *Frame Check Sequence Field*: A cyclic redundancy check (CRC) is used by the receive and transmit algorithms to generate a CRC value for the FCS field. The Frame Check Sequence (FCS) field contains a 4 byte CRC value. This value is computed as a function of the contents of all fields of the frame except the preamble, SFD and FCS.

**c. Invalid MAC Frame**

Any frame which meets the one of the following conditions is an invalid frame:

1. The frame length is inconsistent with the length field.
2. It is not integer number of bytes in length.
3. The bits of incoming frame do not generate a CRC value identical to the one received.

#### **4. Media Access Control**

##### **a. Overview of Functional model**

The MAC sublayer defines a medium-independent facility, built on the medium-dependent physical facility provided by the physical layer, and under the access-layer-independent local area network LLC sublayer. Since the LLC sublayer and the MAC sublayer together are intended to have the same function with the Data Link Layer of the OSI, as in Figure 3 on page 22 the two main functions which are generally associated with a data link control procedure has to be performed in the MAC sublayer. These functions are:

1. Data encapsulation (transmit and receive)
  - a. Framing (frame boundary delimitation, frame synchronization)
  - b. Addressing (handling of source and destination address)
  - c. Error detection (detection of physical medium transmission errors)
2. Media access management
  - a. Medium allocation (collision avoidance)
  - b. Contention resolution (collision handling)

##### **b. The Operation of CSMA/CD**

Transmit frame operations are independent from the receive frame operations. A transmitted frame addressed to the originating station will be received and passed to the LLC sublayer at that station. These characteristics of the MAC sublayer may be implemented by functionality within the MAC sublayer or full duplex characteristics of portions of the lower layers in the forms of transmission and reception:

##### **1. Transmission without contention**

When a LLC sublayer requests the transmission of a frame, the Transmit Data Encapsulation component of the MAC sublayer constructs the frame from the LLC supplied data. It appends a preamble and a Start Frame Delimiter to the beginning of the frame and if it is required it adds the PAD field at the end of the frame. The destination and source addresses, a length count field, and a frame check sequence is also added by MAC sublayer.

##### **2. Reception without contention**

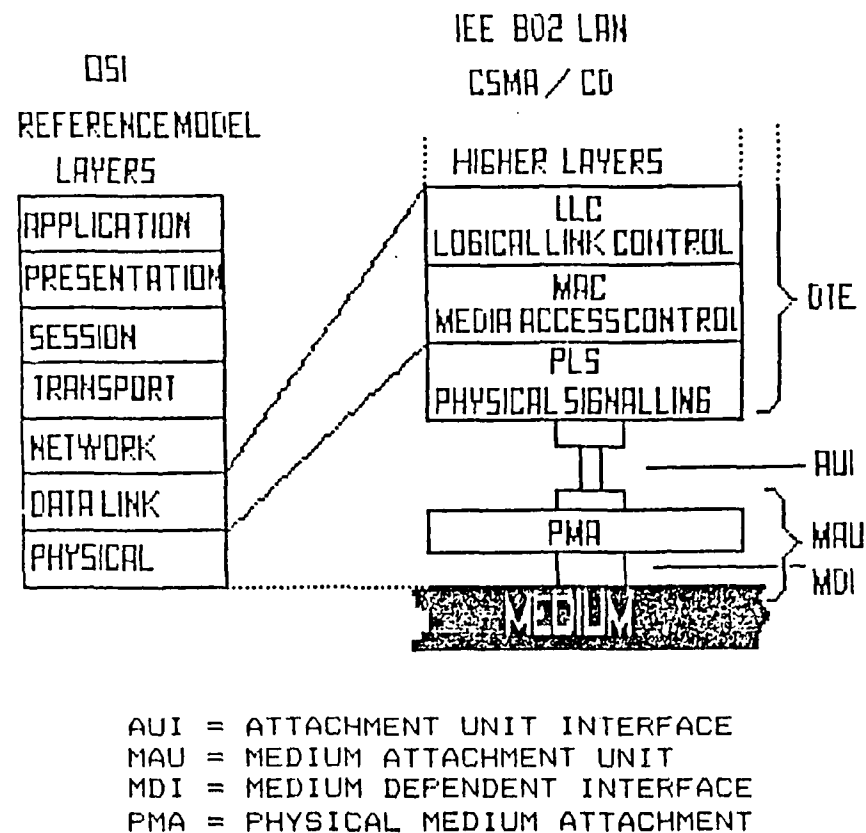


Figure 3. LAN Standard Relationship to the OSI Reference Model

At each receiving station, the arrival of a frame is first detected by the PLS, which responds by synchronizing with the incoming preamble and by turning on the carrier sense signal. As the encoded bits arrive from the medium, they are decoded and translated back into binary data. The PLS passes subsequent bits up to the MAC sublayer, where the leading bits are discarded, up to and including the end of the preamble and Start Frame Delimiter.

Meanwhile, the Receive Media Access Management component of the MAC sublayer collects the bits as long as the carrier signal remains on. When the carrier sense signal is removed, the frame is truncated to an octet boundary, if necessary, and passed to Receive Data Decapsulation for processing.

Receive Data Decapsulation checks the frame's Destination Address field to decide if the frame is received to this station. If so, it passes

the frame to the LLC sublayer checking for invalid MAC frames by inspecting the frame check sequence.

**c. *Access Interference and Recovery***

If multiple stations attempt to transmit at the same time, the overlap of the messages is called a collision. Once the collision occurs, the all other stations can be assumed to have noticed the signal (by way of carrier sense) and to be deferring to it.

In the event of a collision, the transmitting station's Physical Layer initially turns on the collision detect signal. This is then sensed by the Transmit Media Access Management, and collision handling begins. First Media Access Management enforces the collision by transmitting a bit sequence called jam. After that Transmit Media Access Management terminates the transmission and schedules another transmission attempt after a randomly selected time interval. Retransmission is attempted again in the face of repeated collisions. Since repeated collisions indicate a busy medium, however, Transmit Media Access Management attempts to adjust the medium load by backing off (voluntarily delaying its own retransmissions to reduce its load on the medium). This is accomplished by expanding the interval from which the random retransmission time is selected on each successive transmit attempt. Eventually, either the transmission succeeds, or the attempt is abandoned on the assumption that the medium has failed or has become overloaded.

**d. *Relationships to LLC sublayer and Physical Layer***

The MAC sublayer provides services to the LLC sublayer required for the transmission and reception of frames. Although certain errors are reported to the LLC, error recovery is not provided by MAC, and it may be provided by LLC or higher sublayers.

**e. *CSMA/CD Access Method Functional Capabilities***

The following is the summary of the capabilities of the standard:

**1. For frame transmission**

- a. Accepts data from the LLC sublayer and constructs a frame

- b. Presents a bit-serial data stream to the physical layer for transmission on the medium
- 2. For frame reception
  - a. Receives a bit-serial data stream from the physical layer
  - b. Presents to the LLC sublayer frames that are either broadcast frames or directly addressed to the local station
  - c. Discards or passes to Network Management all frames not addressed to the receiving station
- 3. Defers transmission of a bit-serial stream whenever the physical medium is busy
- 4. Appends proper FCS value to outgoing frames and verifies full byte boundary alignment
- 5. Checks incoming frames for transmission errors by way of FCS and verifies byte boundary alignment
- 6. Delays transmission of frame bit stream for specified interframe gap period
- 7. Halts transmission when collision is detected
- 8. Schedules retransmission after a collision until a specified retry limit is reached
- 9. Enforces collision to ensure propagation throughout network by sending jam message
- 10. Discards received transmissions that are less than a minimum length
- 11. Appends preamble, Start Frame Delimiter, DA, SA, length count, and FCS to all frames, and inserts pad field for frames whose LLC data length is less than a minimum value
- 12. Removes preamble, Start Frame Delimiter, DA, SA, length count, FCS and pad field (if necessary) from received frames

#### **5. CSMA/CD Media Access Control Method**

For the precise algorithmic definition of the MAC process the Pascal programming language has been used. However, this model has been chosen for clarity and simplicity and the chosen language does not require that all the procedures will be implemented by a program executed by a computer. The implementation may consist of any appropriate technology including hardware, firmware, software and any combination.

Similarly it shall be emphasized that it is the behavior of any MAC sublayer implementations that shall match the standard, not their internal structure. The timing problems are handled in two ways

1. *Processes Versus External Events.* It is assumed that the algorithms are executed very fast relative to external events, in the sense that a process never falls behind in its work and fails to respond to an external event in a timely manner.
2. *Processes Versus Processes.* Among processes, no assumptions are made about relative speeds of execution. This means that each interaction between two processes shall be structured to work correctly independent of their respective speeds. More detailed information about the implementation can be found in [Ref. 15].



### III. SPECIFICATION OF CSMA/CD PROTOCOL WITH SCM

The CSMA/CD protocol has been specified in form of  $n$  transmitters,  $n$  receivers, and one controller for a network of  $n$  nodes. The controller controls the channel, which connects transmitters and receivers, to each other in both direction.

In reality, each computer in the system has one transmitter and one receiver. When a computer wants to communicate to another computer it sends the message through its transmitter, and the message is received by the receiver part of the destination computer. The transmitter and the receiver parts of the computer resembles the MAC layer of the CSMA/CD protocol. The communication between the MAC sublayer and the other higher layers are not included in this thesis since it is not a part of this specification. The controller resembles the channel in the system, which the machines communicate through and provides better control on the overall work of the system.

The SCM model of CSMA/CD protocol for a network of  $n$  nodes would be defined as  $C = (M, V)$ , where,

$M = \text{set of the machines in the system} = \{\{m_{t_1}, m_{t_2} \dots m_{t_n}\}, \{m_{r_1}, m_{r_2} \dots m_{r_n}\}, m_c\}$

$m_{t_i}$  = is transmitter  $i$  in the transmitter set

$m_{r_i}$  = is receiver  $i$  in the receiver set

$m_c$  = is the controller

$V = \text{The set of shared variables} = \{ \text{Medium}, \text{Ack}(1..n), G(1..n) \}$

The read acces variables for each machine

$R_{t_i} = \{ \text{Medium.Address}, \text{Medium.Data}, G(i) \}$

$$R_{r_i} = \{\text{Medium.Address, Medium.Data, G(i)}\}$$

$$R_c = \{\text{Medium.Address, Medium.Data, G(1..n), Ack(1..n)}\}$$

The write acces variables for each type of machine

$$W_{t_i} = \{\text{Medium.Address, Medium.Data, G(i)}\}$$

$$W_{r_i} = \{\text{G(i), Ack(i)}\}$$

$$W_c = \{\text{Medium.Address, Medium.Data, G(1..n), Ack(1..n)}\}$$

S = sets of states of machines =  $\{S_{t_i}, S_{r_i}, S_c\}$

$$S_{t_i} = \text{states of } m_{t_i} = \{0, 1, 2, 3\}$$

$$S_{r_i} = \text{states of } m_{r_i} = \{0, 1, 2\}$$

$$S_c = \text{states of } m_c = \{0, 1, 2\}$$

Initial state(s) of each type of machine  $m_{t_i} = m_{r_i} = m_c = 0$

The names dedicated to each type of machine

$$N_{t_i} = \{\text{Pkt\_Queued, Xmit, OK, Collision, Retry}\}$$

$$N_{r_i} = \{\text{Clean, Collision, Receive, Return}\}$$

$$N_c = \{\text{Clean, Garbage, Message, Reset}\}$$

The local variables for each type of machine

$$L_{t_i} = \{\text{Msg.DA, Msg.Data}\}$$

$$L_{r_i} = \{\text{Inbuffer}\}$$

$$L_c = \{\text{Medium.Address, Medium.Data, G}(1..n), \text{Ack}(1..n)\}$$

$\tau_{t_i}$ ,  $\tau_{r_i}$  and  $\tau_c$  partial transition functions from the states and the names of a machine type to the states of same type.

$\tau_{t_i}$	$\tau_{r_i}$	$\tau_c$
0 × Pkt_Queued $\mapsto 1$	0 × Receive $\mapsto 1$	0 × Message $\mapsto 1$
1 × Xmit $\mapsto 2$	0 × Collision $\mapsto 2$	0 × Garbage $\mapsto 2$
2 × OK $\mapsto 0$	1 × Return $\mapsto 0$	1 × Reset $\mapsto 0$
2 × Collision $\mapsto 3$	2 × Clean $\mapsto 0$	2 × Clean $\mapsto 0$
3 × Retry $\mapsto 1$		

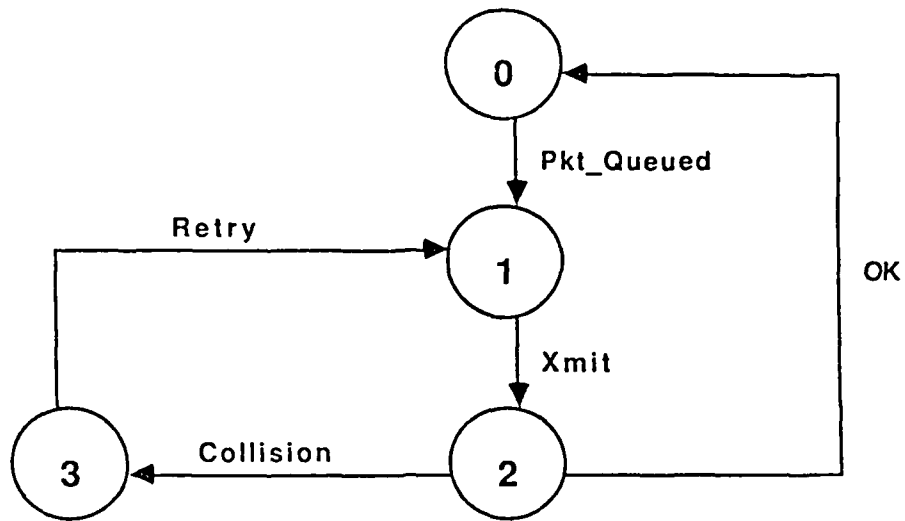
The specification of this model is shown in Figure 4 on page 29 (transmitter), Figure 5 on page 30 (receiver), and Figure 6 on page 31 (controller) with their predicate\_action tables.

The system works as follows:

The initial states of all three machines are 0, and the shared variables  $G(1..n)$  and  $\text{Ack}(1..n)$  are set to 0. The shared variable *Medium.Addr*, *Medium.Data*, and the local variables *Msg.DA*, *Msg.Data* and *Inbuffer* are empty. The local variable *Msg.Da*, *Msg.Data* and *Inbuffer* are used to store the data blocks which are transferred from  $m_1$  to  $m_2$  through the shared variables *Medium.Addr* and *Medium.Data*.

1. When the computer has a message to send, it passes the message to its transmitter part. The transmitter takes this message writing its data part onto its local variable *Msg.Data* and address part onto its local variable *Msg.DA*, and changes its state from state 0 to state 1 executing the transition *Pkt\_Queued*. Here, at state 1 the transmitter waits if the *Medium* is not empty.
2. As soon as the *Medium* becomes available, the transmitter executes the transition *Xmit*, copies the *Msg.DA* variable to the shared variable *Medium.Addr* and *Msg.Data* variable to the *Medium.Data*, and changes its state to state 2 and waits.

# TRANSMITTER



MSG :      DA              DATA

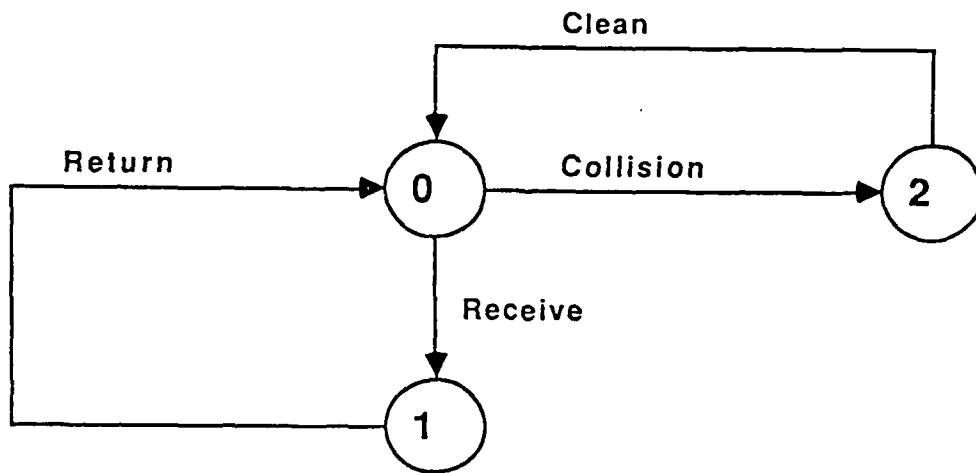


i : MA    (My Address)

TRANSITION	EN.PREDICATE	ACTION
Pkt_Queued	Msg <> E	-
Xmit	Medium = E	Medium := Msg
Collision	Medium = Garbage	G (i) := 1
Retry	G (i) = 0	-
OK	Medium = E	Msg := E

Figure 4. Specification of the Transmitter

## RECEIVER



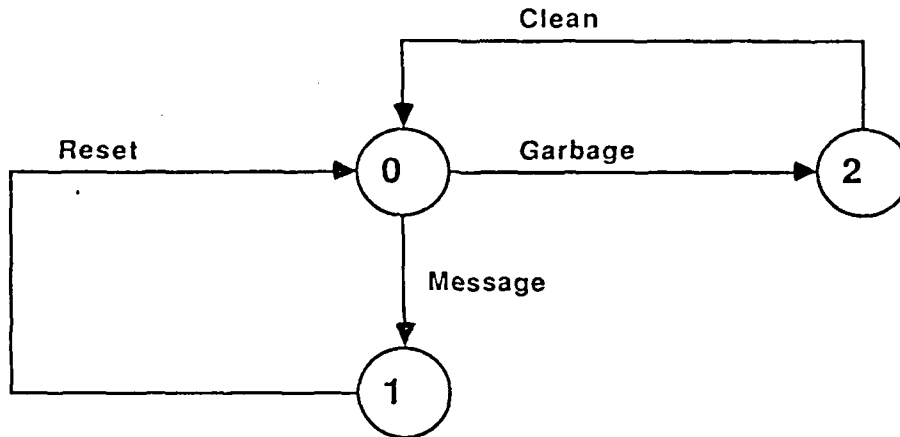
INBUFFER : DATA



TRANSITION	EN.PREDICATE	ACTION
Receive	Medium.Addr = MA	Inbuffer:=Medium.Data
Collision	Medium = Garbage	G(i) := 1
Return	T	Ack(i) := 1
Clean	G (i) = 0	-

Figure 5. Specification of the Receiver

# CONTROLLER



MEDIUM : ADDR. DATA



ACK(1..n) :



G(1..n) :



TRANSITION	EN.PREDICATE	ACTION
Message	Medium <> ( G V E )	-
Garbage	Medium = G	-
Clean	G(1..n) = 1	G(1..n) := 0 Medium := E
Reset	Ack(medium.addr)=1	Ack(1..n) := 0 Medium := E

Figure 6. Specification of the Controller

3. The controller executes the transition *Message* and moves to state 1 from state 0, since the conditions for the enabling predicate of this transition are true.
4. The receiver checks the *Medium.Addr* and if the address is its address, it executes the transition *Receive* and moves to state 1 copying the *Medium.Data* to its local variable *Inbuffer*. Here if the message has been copied without errors, it changes its state from state 1 to state 0 executing the transition *Return* and changes the value of the shared variable *Ack(2)* to 1.
5. Since the 1 valued *Ack(2)* is an enabling predicate of the transition *Reset*, the controller empties the variables *Medium.Addr* and *Medium.Data*, and gives 0 value to the *Ack(1..n)* and moves to state 0.

The situation described above is a transmission without collision and in CSMA/CD, once a transmitter begins to transmit none of the other transmitters attempt to transmit, since the enabling predicates of the protocol system do not allow this transmission.

However, when the medium is available, two or more transmitters may try to transmit simultaneously and this results in a collision which is considered to be an ill\_defined transition in SCM.

It should be noted here that the collision of two or more messages can be modeled with SCM as a simultaneous write to a shared variable, because simultaneous transitions are allowed in this model. However simultaneous transitions can not be modeled with pure CFSM. Further CFSMs do not allow shared variables, and the machines of this model use only the channels dedicated to themselves in form of FIFO queues, so the bus is also difficult or impossible to model.

The colliding messages leave a garbled message in both shared variables *Medium.Addr* and *Medium.Data*. At this system state both transmitters are in state 2 and the receiver and the controller are in state 2. This garbled message is an enabling predicate for the controller and all other machines. Under this circumstances the system works as follows:

1. The controller changes its state to state 2, executes the transition *Garbage*.
2. The receiver moves to state 2 and executes the transition *Collision*, makes the value of shared variable  $G(2) = 1$ .

3. The transmitter executes the transition *Collision* and changes the value of  $G(1)$  to 1 and goes to state 3.
4. When the values of shared value  $G(1..n)$  becomes 1 for all the addresses in the system, the controller empties the medium (cleans), changes the values of  $G(1..n)$  to 0, and goes to state 0.
5. The receiver changes its state from state 2 to state 0 executing the transition *Clean*, and waits for the new message to come.
6. The 0 valued  $G(1)$  enables the transition *Retry* of the transmitter, and the transmitter goes back to state 1 and reattempts to send its message to the destination address again since the first attempt has been ended with a garbled message.

This sequences continue in a cyclic behavior if any of the machines in the system has a message to send.



#### IV. SYSTEM STATE ANALYSIS OF THE SYSTEM

The *system state analysis* is the analysis method of SCM in order to find some possible errors of the specification, such as *deadlock*, *unspecified reception* and *non-executable transition*. The tuple used in this analysis is called as *system state* and it is the tuple of all machine states together with the enabled outgoing transitions.

The system state analysis of the model has been done for a system with two transmitters. Since a one transmitter system never ends up with a collision, an analysis with two transmitters is a more realistic analysis than analysis of a single transmitter system. This analysis can also be done for any finite number of transmitters. However, since the increasing number of machines increases the number of states in the analysis exponentially it becomes impractical to generate an analysis for many machines.

##### A. ANALYSIS

In two transmitter specification, both transmitters have the same specifications except their addresses ( in reality they are the transmitters of two different computers in the system ) and they both try to transmit to the same receiver ( to a third computer in the system ), either at different times or simultaneously.

The transmission may result with a collision or with a successful transmission. All these possible situations have been tried during the analysis. However some transitions which are clearly harmless ( i.e., when one of the transmitters has been using the Medium, the other transmitter , which is in state 0 may change its state to state 1, because of the data send request of the higher layer) are omitted and the values of the local and shared variables has not been shown in the system state tuples to make the analysis more clear.

The machines of the system are defined as a tuple (*transmitter\_1*, *transmitter\_2*, *receiver*, *controller*). For an example, the state tuple ( 1,2,0,1 )

defines a a system state where the *transmitter\_1* is in state 1 (it has a message to send), the *transmitter\_2* is in state 2 (it is transmitting a message already), the *receiver* is in state 0 (its all local variables are empty), and the *controller* is in state 1 (the medium has a written message in it). The transitions of these machines are labeled with appropriate numbers to make clear the owner of the transition.

Initially all of the machines are in their initial states ( 0,0,0,0 ), and the variables *Msg.DA*, *Msg.Data*, *Inbuffer*, *Medium.Addr* and *Medium.Data* are empty, and *Ack*(1..*n*), *G*(1..*n*) valued with 0. When a transmitter has a message to send it moves to state 1. This transition may be taken by one transmitter or both. These transitions result with following possible three system states; ( 0,1,0,0 ), ( 1,0,0,0 ), ( 1,1,0,0 ).

From the initial state if one of the transmitters begins to transmit this is considered as a successful transition, since the enabling predicates of the specification prevents the access of the other transmitter (or transmitters in a multi\_transmitter system) to the Medium.

For an example in state tuple ( 1,1,0,0 ), both transmitters have messages to send. But once the *transmitter\_2* begins to transmit and changes its state to state 2, the system state becomes

( 1,2,0,0 )

and *transmitter\_1* waits until the channel becomes available again. In this state there are two transitions available. The *controller* may change its state to state 1 since ther is a message in the Medium or the *receiver* may change its state to state 1 since the destination address of the message is its address in our two transmitters, one receiver model. In first case the system state becomes

( 1,2,0,1 )

For the second case the system state is

$$(1, 2, 1, 0)$$

The next state after the state  $(1, 2, 0, 1)$  is the state

$$(1, 2, 1, 1)$$

since the *receiver* receives the message and changes its state to state 1. The system state

$$(1, 2, 0, 1)$$

follows this state. It should be noted here, while this system state seems to be the same state with one of the system states above, it is quite different than the other since the transition *Return* changes the value of the shared variable  $Ack(i)$  to value of 1. The system state following this state is

$$(1, 2, 0, 0)$$

since the *controller* changes its state to state 0 cleaning the shared variables  $Ack(1..n)$  and *Medium*. The system state following this state is the state

$$(1, 0, 0, 0)$$

where the the shared variable *Medium* is available for the transmission of *transmitter\_1*.

But the system state  $(1, 1, 0, 0)$  may also result with a collision, because of the simultaneous attempts to transmit. After the garbled medium cleaned, the model returns to one of its initial state defined above, and another transition scheme begins if there is any. The detailed system state analysis of the

system is as in Figure 7 on page 38, Figure 8 on page 39 and Figure 9 on page 40.

## **B. CALCULATION OF THE NUMBER OF ANALYSIS STATES**

While the specification of the system includes one transmitter, one receiver and one controller, and the system state analysis of the model has been done for two transmitters, one receiver and one controller, in reality LANs carry many more machines. A system state analysis of the model with multiple machines will end up with a very large number of the states, depending the number of machines in the system. It is probably useful to know the number of states in the analysis before beginning to the analysis.

In order to come up with a formula which finds the number of states in the analysis, the state analysis is divided in to 7 abstract parts as it can be seen in Figure 10 on page 41.

Part 1 of the figure includes beginning states of the analysis and they are called *entrance states*. In these states only the transmitters can be in one of the two states; the state 0 or the the state 1, thus they may or may not have messages to send in their local variables, and the other machines ( the receiver and the controller ) can only be in state 0, since there is no message in the Medium. So the system state tuples of these states are between ( 0,0,...,0,0 ) and ( 1,1,...0,0 ). Here the last two digits denotes the states of receiver and controller and they can only be in state 0 at the entrance states. Calculation of the number of states in part 1 is:

$n$  = Number of transmitters in the system

$S_1$  = Number of entrance states in the system

$$S_1 = 2^n$$

Parts 2,3,4 and 5 all include the same number of states and has 7 states for multi\_transmitter single\_receiver single\_controller system and they are called

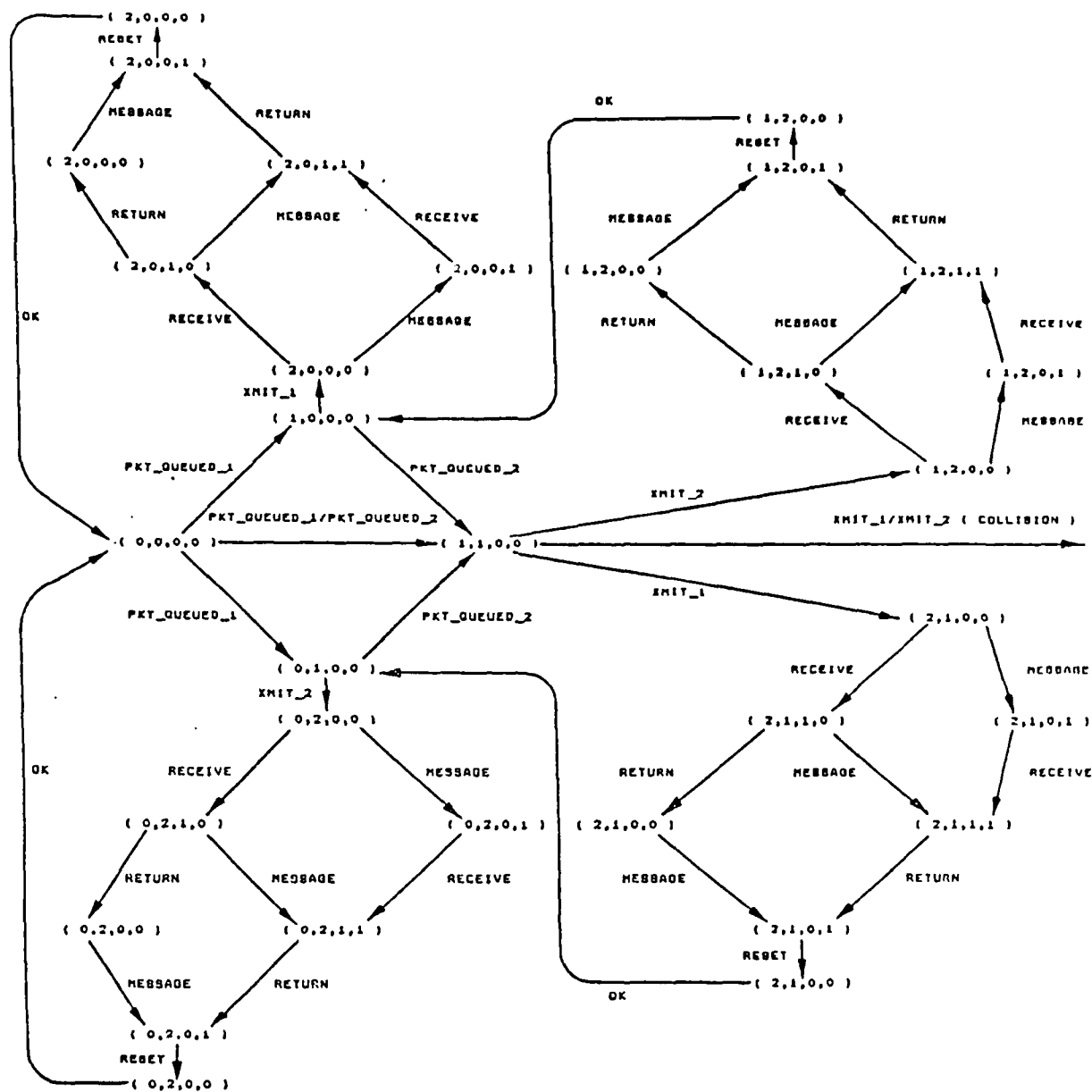


Figure 7. System State Analysis (continues)

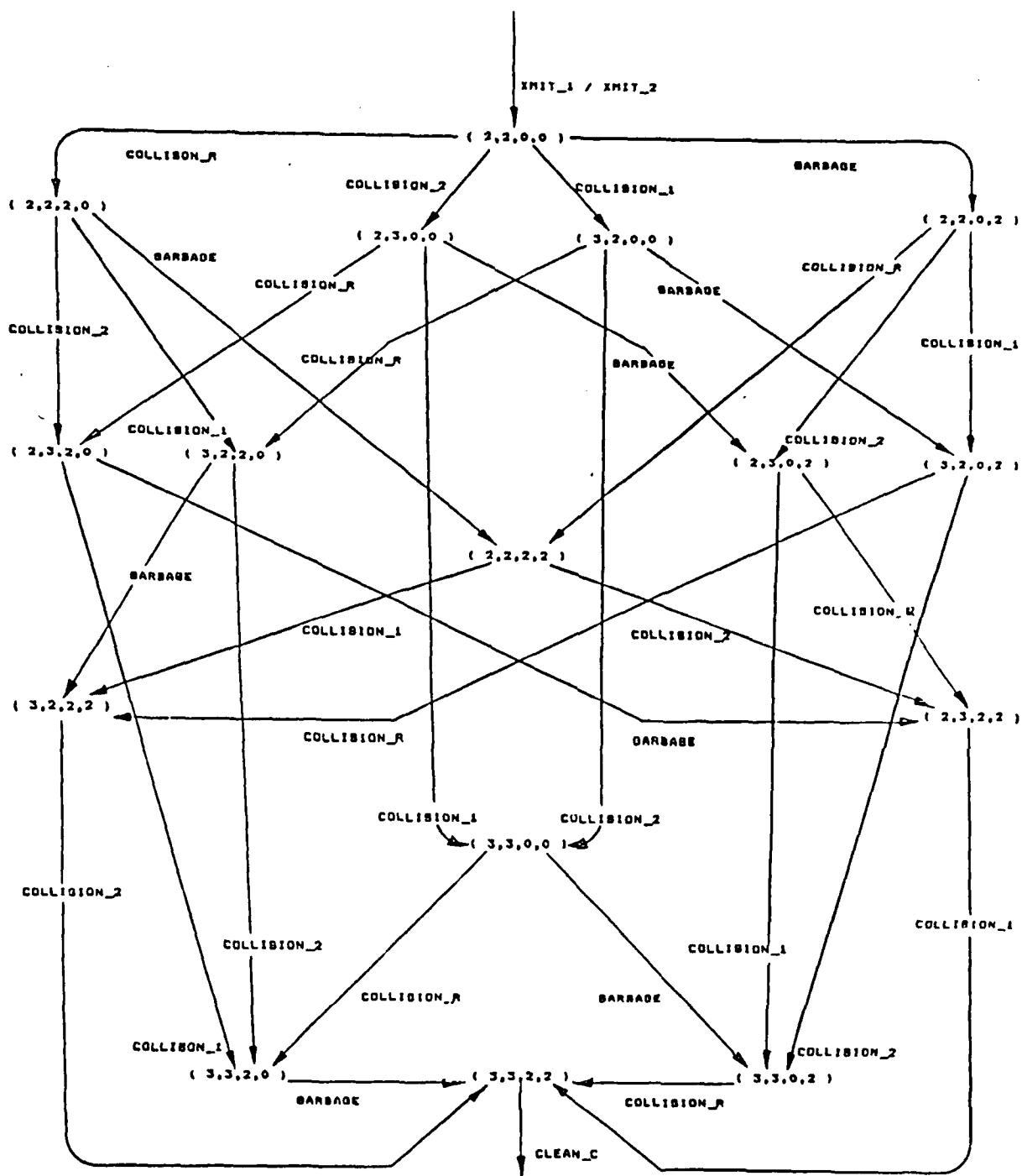


Figure 8. System State Analysis (continues)

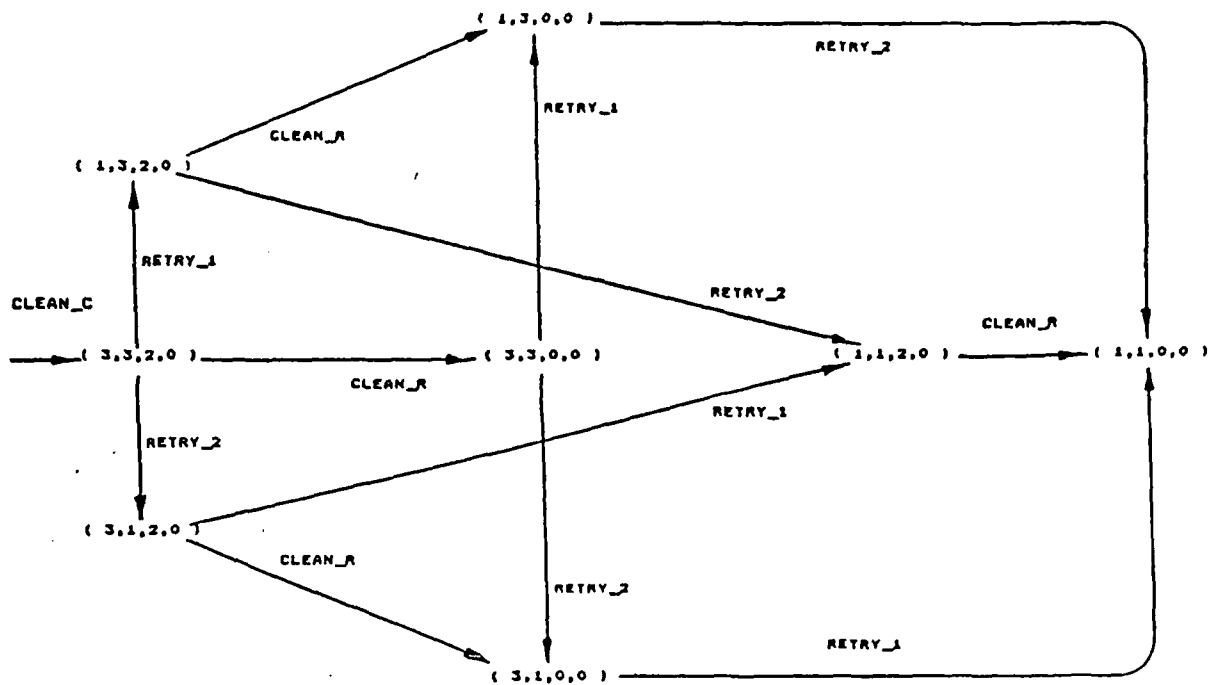


Figure 9. System State Analysis

successful transmission states. However it is almost impossible to give a formula which calculates the number of the states in a multi\_transmitter-multi\_receiver system in this part because of the repeating states of analysis which are seem exactly same, but a lot different in *global state analysis* because of the different values of local and shared variables. But it is quite possible to calculate the number of system states which may end up with a successful transmission.

The number of system states which will result with a succesful transmission begins with the transition Xmit and the transmitter changes its state from state 1 to state 2. So any system state where  $i$  ( $1 \leq i \leq n$ ) transmitters are in state 1 may result with  $i$  attempts to transmit. Since we can choose the number of transmitters in state 1 with different combinations in an  $n$  transmitter system with:

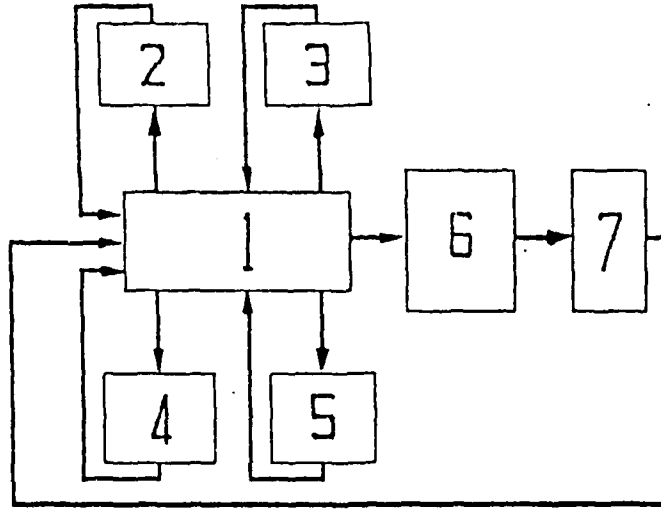


Figure 10. Abstract Modeling of System State Analysis

$$\sum_{i=1}^n \frac{n!}{(n-i)!i!}$$

and from any of these states the  $i$  transmitters may attempt to transmit in  $i$  different combinations where each may end up 7 states of succesful transmission. Thus, if:

$$S_{2+3+4+5} = \text{Number of states in parts 2, 3, 4 and 5}$$

$$S_{2+3+4+5} = \sum_{i=1}^n \frac{n!}{(n-i)!i!} \times 7i$$

The states where more two or more transmitters are in state 1 may lead to collisions because of simultaneous attempts to transmit. The number of transmitters involved in the collision may be any number between two and  $i$  if  $i \geq 2$  and in these system states, while the transmitters involved in the collision



are strictly in state 2, the other transmitters may be in state 0 or state 1. Thus, if

$c$  = number of states involved to the collision

The other states may be in  $2^{(n-c)}$

Since the number of transmitters which are involved in the collision should be at least two, we can find the number of states which may end up with collisions:

$$\sum_{c=2}^n \frac{n!}{(n-c)!c!} \times 2^{(n-c)}$$

Since only the transmitters involved to the collision, and the receiver and the controller deal with the situation and the transmitters may be in state 2 or state 3 while the receiver and controller may be in state 0 or state 2, the number of states in part 6 (*collision states*):

$S_6$  = Number of states in part 6

$$S_6 = \sum_{c=2}^n \frac{n!}{(n-c)!c!} \times 2^{(n-c)} \times 2^{(c+2)}$$

The states in part 7 (*cleaning states*) begin with the state ( 3,3,...,2,0 ) and end with one of the entrance system states. Since the transmitters may be in state 0 or state 1, and the receiver may be in state 2 or state 0, Thus if we name:

$S_7$  = Number of states in part 7

$$S_7 = 2^{(c+1)} - 1$$

So, if "S = Number of states in a system state analysis" the formula which finds the number of states in a system analysis which analyses a protocol with multi\_transmitter, single receiver and single controller:

$$S = S_1 + S_{2+3+4+5} + S_6 + S_7$$

$$S = 2^n + \sum_{i=1}^n \frac{n!}{(n-i)!i!} \times 7i + \sum_{c=2}^n \frac{n!}{(n-c)!c!} \times 2^{(n+2)} + 2^{(c+1)} - 1$$

This formula clearly shows that a system state analysis for more transmitters will end up with more system states.

## V. CONCLUSIONS AND SUGGESTIONS FOR FURTHER WORK

In this thesis a different specification of the IEEE Standard 802.3 CSMA/CD has been provided. The protocol modeling method used for this specification is *systems of communicating machines*, and this new specification of the standard has been kept as close to the original as possible. Some details of the protocol have been omitted which are irrelevant to the purposes of the thesis. This model provided in the thesis represents the general model of the machines in an  $n\_node$  network. The only difference between the various machines in the network are their network addresses. So this specification may be used to specify a network with any number of nodes.

The *system state analysis* of this specification has shown that, the specification is free from *deadlock* and some other kind of *errors*. In order to provide a more realistic analysis which also analyses the *collision* modeling of the specification, the model used is a *two\_transmitters, one receiver* and *one controller*. While it is possible to analyse the specification for more transmitters and receivers, it is impractical to handle this kind of analysis in the limits of the thesis. The formula given with the analysis of the specification is used to calculate the number of states in a *multi\_transmitters, one receiver* network.

It should be noted that under the formal specification of CSMA/CD protocol longer delay times may be expected because of the heavy traffic load of the system. While CSMA/CD has proven its usefulness as a LAN specification, in the future an increase in the number of stations may cause unacceptable delays in the system. However, since the specification has been done to provide fair access to all devices in the system and since its structure is distributed, this protocol will probably need a new specification in order to serve to the LANs with larger numbers of stations.

It should also be noted that the specification provided in this thesis does not support the distributed structure since there is a controller in the system.

However the controller of the system does not provide full control over the stations like other centralized structures (i.e., there is no medium access request for any station) and it is fair to the all of the stations. This is a structure which handles dissadvantages of both structures and may provide an important achievement over the existing protocol and challenge with longer delay times of crowded LANs.

Potential research subjects in this area include the following:

1. Improvement of the existing protocol, or a new protocol which handles collisions better and overcomes the expected longer delay times.
2. Improvement of the specification of this thesis, adding collision backup time and other MAC sublayer tasks to the specification.
3. An analysis of this specification for broadcast addresses in a multi\_receiver system.

## LIST OF REFERENCES

1. Martin, J., *Local Area Networks*, New Jersey : Prentice Hall, 1989.
2. Ayik, N., *An Analysis of the Token Ring Protocol as Specified in ANSI/IEEE Standard 802.5-1985*, Master's Thesis, Naval Postgraduate School, Monterey, CA, March 1989.
3. Halsall, F., *Data Communications, Computer Networks and OSI*, England : Addison-Wesley, 1988.
4. Stallings, W., *Data and Computer Communications*, New York : Macmillan, 1988.
5. Lundy, G.M., *Systems of Communicating Machines : A Model for Communication Protocols*, Ph.D. Thesis, School of Information and Computer Science, Georgia Institute of Technology, Atlanta, GA, 1988.
6. Rudin, H., "An Informal Overview of Formal Protocol Specification," *IEEE Communications Magazine*, v.23 No.3, pp.46-52, March 1985.
7. Lundy, G.M., Notes, presented in the Formal Modeling of Computer Networks course, Naval Postgraduate School, Monterey, CA, January-March 1989.
8. Miller, R. E., "The Communicating Finite Machine Model for Communication Protocols," paper, presented in the Formal Modeling of Computer Network Protocols course, Naval Postgraduate School, Monterey, CA, January-March 1989.

9. Castanet, R., Dupuex, A., and Guitton, P. "Ada, a Well-suited Language for the Specification and Implementation of Protocols," *Proceedings of the Fifth International Workshop on Protocol Specification, Testing and Verification*, Toulouse-Moissac, France, June 10-13, 1985.
10. Fleischmann, A., *PASS : A Technique for Specifying Communication Protocols*, Protocol Specification, Testing and Verification VII, North-Holland, 1987.
11. Linn, R. J., "The Features and Facilities of Estelle : A Formal Description Technique Based Upon an Extended Finite State Machine Model," *Proceedings of the Fifth International Workshop on Protocol Specification, Testing and Verification*, Toulouse-Moissac, France, June 10-13, 1985.
12. Institute of Electrical and Electronics Engineers, *ANSI/IEEE Std.802.5-1985, Token Ring Access Method and Physical Layer Specifications*, 1985.
13. Lundy, G.M., and Miller, R.E., *Specification and Analysis of a Data Transfer Protocol Using Systems of Communicating Machines*, School of Information and Computer Science, Georgia Institute of Technology, Atlanta, GA, October 1, 1988.
14. Miller, R.E., and Lundy, G.M., "An Approach to Modeling Communication Protocols Using Finite State Machines and Shared Variables," *IEEE Global Telecommunications Conference*, Houston, TX, December 1-4, 1986.
15. Institute of Electrical and Electronics Engineers, *ANSI/IEEE Std.802.3-1985, Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, 1985.

# INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2. Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3. Deniz Kuvvetleri Komutanligi Kutuphane, Bakanliklar, Ankara, TURKEY	1
4. Deniz Harp Okulu Komutanligi Kutuphane, Tuzla. ISTANBUL, TURKEY	1
5. Department of Computer Science Naval Postgraduate School, Attn: Prof. G.M. Lundy, Code 52Ln Monterey, CA 93943-5000	5
6. Department of Computer Science Naval Postgraduate School, Attn: Prof. T. Wu, Code 52Wq Monterey, CA 93943-5000	1
7. Mehmet Nuri LOFCALI Yurt mah. 652 sk, 01140 Adana, TURKEY	2
8. Ortadogu Teknik Universitesi Kutuphane, Ankara, TURKEY	1
9. Bogazici Teknik Universitesi Kutuphane, Istanbul, TURKEY	1